1/24/2025

Course Project: Client/Server Application

CSI 2470/Winter – 2025

Instructor: Dafer Alali
CSI 2470/OAKLAND UNIVERSITY

I. **Course:** 2470 Introduction to Computer Networks

II. **Project Title:** Building a Client/Server Application

III. **Objective:**

The objective of this project is to understand the client-server architecture by implementing a networked application that communicates using sockets. You will apply concepts such as TCP/UDP communication, server and client handling, and basic data transfer protocols.

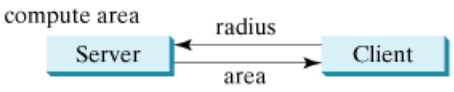
IV. Project Description:

You are tasked with creating a **Client-Server** application that allows communication over a network. The server will accept multiple client connections, and clients will send data to the server. The server should process the data and respond to the clients. You may implement either TCP or UDP for your communication protocol. The server should clearly describe what kind of **services** provide for the clients. (See Figure A)

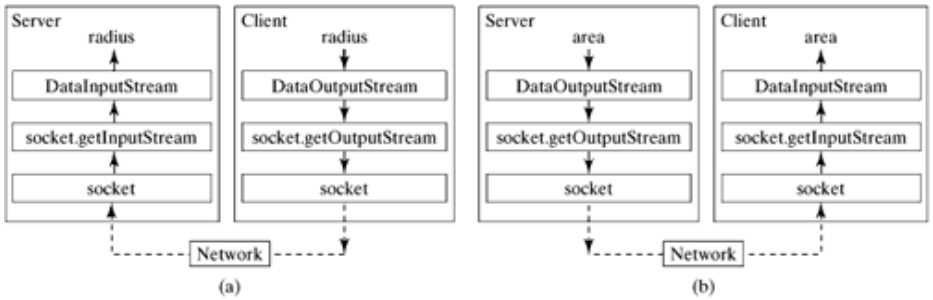
The project will include:

- Socket programming
- Handling multiple client connections (for TCP)
- Request/response protocols
- Error handling and basic security measures

The client sends the radius to the server; the server computes the area and sends it to the client.

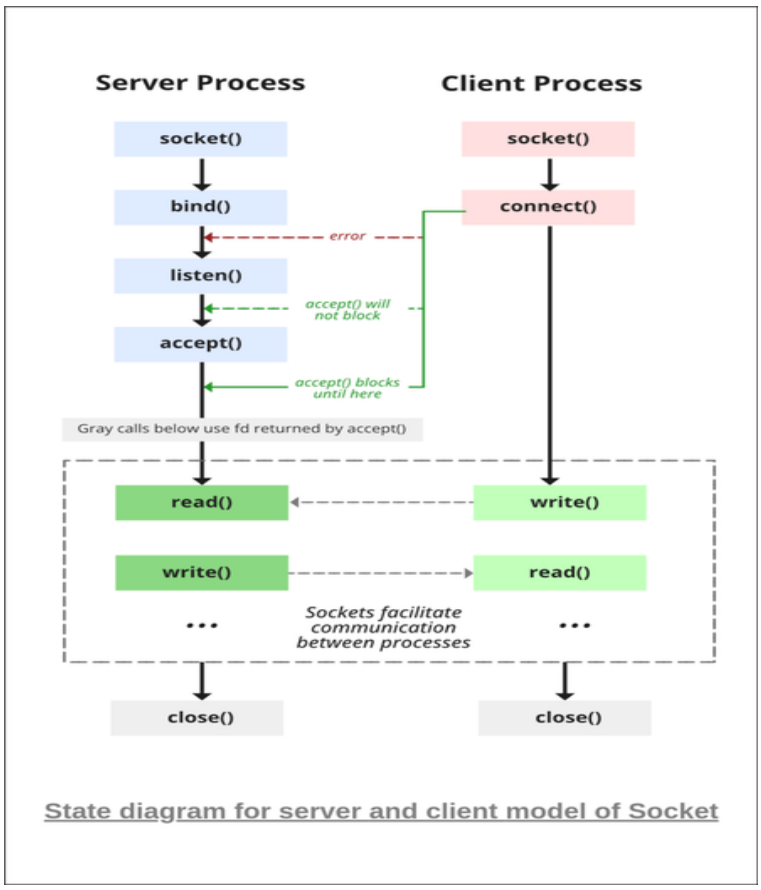


(a) The client sends the radius to the server. (b) The server sends the area to the client.



V. Technical Requirements:

1. **Programming Language**: You may use C, C++, Python, or Java. Ensure your program runs on a standard operating system (Linux, Windows, or macOS).
2. **Networking Protocol**:
    - Choose between **TCP** (Transmission Control Protocol) or **UDP** (User Datagram Protocol).
    - For TCP: Implement a simple chat or file transfer system.
    - For UDP: Create a lightweight application, such as a basic online game or a real-time status checker.
3. **Features**:
    - **Client Process**:
        - Connect to the server using an IP address and port.
        - Send messages or data to the server.
        - Receive responses from the server.



State diagram for server and client model of Socket

- **Server Process**:
  - Listen on a specific port.
  - Accept multiple client connections (for TCP).
  - Process the client's request and respond.
  - Optionally, log client connections and communications.

4. **Concurrency** (TCP): The server must be able to handle multiple clients simultaneously using threads or asynchronous programming techniques.
5. **Error Handling**: Implement basic error handling for situations like lost connections, invalid inputs, or server overloads.
6. **Security**: Implement at least one security feature. Examples include:
   - Input validation to prevent buffer overflow attacks.
   - Basic authentication (password-protected connection).

VI. Deliverables:

1. **Code**:
    - Client and server source code.
    - Include a **README** file with instructions on how to compile and run the application.
2. **Report**:
    - Explain the architecture and design choices.
    - Describe the communication protocol you chose and why.
    - Discuss any issues or challenges you encountered and how you solved them.
3. **Presentation**:
    - You will be required to demonstrate your application in class by creating professional PPT Slides, and each team should present their works to the class.
    - Show how the client and server communicate over a network.

VII. Grading Criteria:

Component	**Points**
Correct implementation of client-server communication	**20**
Handling multiple clients (TCP) or proper UDP use	**15**
Security and error handling	**15**
Code quality and documentation	**15**
Final report	**20**
Presentation/demonstration	**15**
Total	**100**

VIII. Important Dates and Time:

- **Final Submission Deadline**: Due on Wednesday April 2nd, 2025, 11:59 PM
- **Presentation Date**: Due Wednesday November April 9th, 2025- Wednesday November April 16th, 2025. Based on the presentations schedule for each group as the following:

	Group members	Group name	Presentation Date
1.		Group 1	April 9th
2.		Group 2	April 9th
3.		Group 3	April 9th
4.		Group 4	April 9th
5.		Group 5	April 9th
6.		Group 6	April 14th
7.		Group 7	April 14th
8.		Group 8	April 14th
9.		Group 9	April 14th
10.		Group 10	April 14th
11.		Group 11	April 16th
12.		Group 12	April 16th
13.		Group 13	April 16th

IX. Additional Resources:

- Tutorials on Socket Programming:
  - Python: https://docs.python.org/3/howto/sockets.html
  - C/C++: https://www.geeksforgeeks.org/socket-programming-cc/
  - Java: https://docs.oracle.com/javase/tutorial/networking/sockets/

X. Submission Instructions:

- Submit your code and report via Moodle by the deadline.
- Ensure your project is well-documented and includes clear instructions on how to run your code.
- **Please note: Your server and the services it provide must be selected from the options available on Moodle.**